Marching Cubes: una implementación eficiente que contempla todos los casos

A. Silvetti, S. Castro y C. Delrieux
Dpto. de Ciencias de la Computación
Universidad Nacional del Sur - Avda. Alem 1253 - 8000 - Bahía Blanca
0291-4595135 - FAX 0291-4595136
{silvetti,uscastro, usdelrie}@criba.edu.ar

Resumen

Se presenta un análisis exhaustivo de las simetrías geométricas involucradas en el algoritmo de *Marchin Cubes*, la cual resuelve todos los problemas de ambigüedades y agujeros de una manera eficiente y sencilla. Las triangulaciones necesarias son 30, y se muestra que otras soluciones propuestas recientemente tienen casos supérfluos y no resuelven completamente el problema.

1. INTRODUCCIÓN

El algoritmo de *Marching Cubes* [1] es una técnica para la extracción de isosuperficies a partir de un conjunto volumétrico de datos, que examina cada elemento de un volumen (celda) y determina la topología de una superficie que pasa por dicho elemento mediante la disposición de los valores de los vértices, según estén por encima o por debajo de un valor umbral o valor de interés especificado por el usuario. Para ese valor, habrá algunas celdas completamente internas o externas a la isosuperficie resultante y valores para los cuales hay que analizar de qué manera la intersectan. El algoritmo produce un conjunto de uno o más polígonos triangulares que se toma como dato de entrada para el algoritmo de rendering. Cada arista de la celda puede tener a lo sumo una intersección con la isosuperficie y ésta se produce cuando el valor umbral queda acotado inferior y superiormente por los valores en los vértices de la arista. La idea original de Lorensen [1] es que se pueden encontrar clases de equivalencia entre combinaciones de valores y triangulaciones. Originalmente eran sólo 15, lo cual daba un algoritmo veloz y sencillo [3]. Sin embargo, pronto se encontraron dificultades respecto de esta primera solución [2].

El primer paso del algoritmo es considerar cada vértice de la celda determinando si el valor es mayor o menor que el valor umbral y asignar entonces al vértice un valor 0 o 1 respectivamente. Si todos los vértices son 0 o 1, el elemento no es atravesado por una isosuperficie y el algoritmo continúa con la siguiente celda. Para los casos restantes, el elemento contendrá una o más caras de la isosuperficie. Los vértices de la celda están numerados secuencialmente de forma tal de poder formar un número binario n cuyo bit más significativo corresponda al seteado para el vértice con mayor número de orden. Este número binario pertenecerá al conjunto [0..255] (8 vértices). El segundo paso del algoritmo consiste en utilizar el número n armado para acceder a una tabla predefinida que enumera cuántos y cuáles triángulos conforman el segmento de isosuperficie que pasa por la celda, qué aristas de la misma contienen los vértices de los triángulos y en qué orden (ver Figuras 1 y 2).

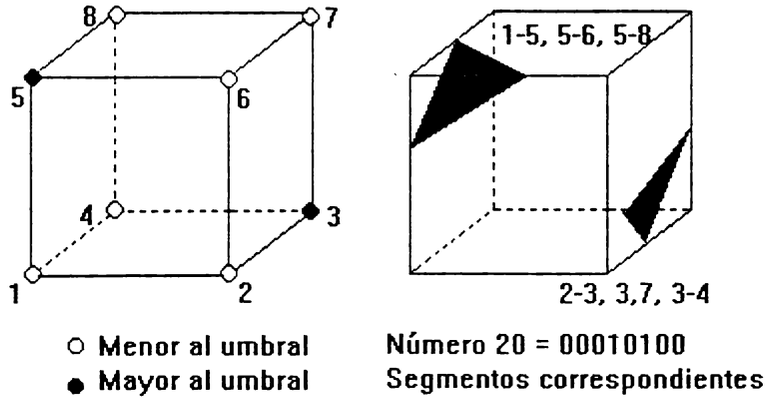

Figura 1: numeración de vértices - Triangulación para el binario 00010100

Al mismo tiempo, es posible guiar la recorrida de las celdas con un algoritmo recursivo utiliando la misma tabla, de manera de evitar una búsqueda exhaustiva. Finalmente, se debe computar la locación exacta de cada segmento de isosuperficie sobre las aristas de la celda usando interpolación lineal del valor umbral entre los valores de los vértices de la misma para dar como resultado los triángulos que conforman la superficie que intersecta a la celda. Dichos triángulos son luego renderizados por medio de las técnicas usuales en Computación Gráfica [3]. La búsqueda de simplificaciones en los casos posibles se basa en simetrías geométricas, las cuales nunca fueron reseñadas exhaustivamente. En la primera implementación se pensaba que los 15 casos reseñados serían suficientes, pero luego se encontraron problemas no triviales originados en situaciones ambiguas.

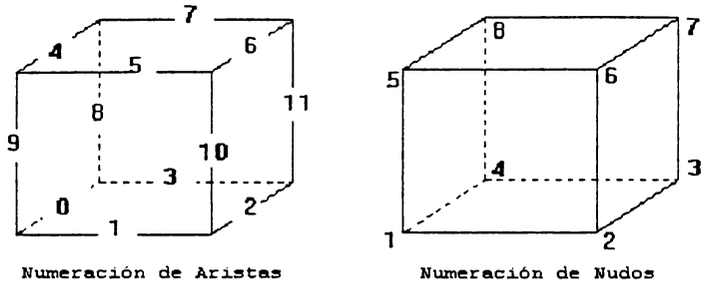

Figura 2: Una celda con sus nudos y aristas numerados. Dicha celda puede ubicarse de 24 formas diferentes, cuatro formas por cada cara de la celda.

2. CLASES DE EQUIVALENCIA Y POSICIONES DE ANÁLISIS

La Figura 3 muestra la triangulación 1 del algoritmo original de Marching Cubes. Utilizando las dos matrices presentadas, podemos encontrar la clase de equivalencia a la que llamaremos *Grupo1* que estará formada por los números 1, 2, 4, 8, 16, 32, 64, y 128 que son todos los números binarios que pueden formarse con un punto marcado (densidad mayor al umbral) en la posición 1 de la celda. De la misma manera, podemos hacer con cada una de las 14 configuraciones del algoritmo original de Marching Cubes. Es decir, hay 14 clases de equivalencia a las que llamamos *Grupo n* $\{1 \leq n \leq 14\}$.